# Optimized Execution of Action Chains through Subgoal Refinement*

**Freek Stulp and Michael Beetz**
Intelligent Autonomous Systems Group, Technische Univerität München
Boltzmannstrasse 3, D-85747 Munich, Germany
{stulp,beetz}@in.tum.de

### Abstract

In this paper we propose a novel computation model for the execution of abstract action chains. In this computation model a robot first learns situation-specific performance models of abstract actions. It then uses these models to automatically specialize the abstract actions for their execution in a given action chain. This specialization results in refined chains that are optimized for performance. As a side effect this behavior optimization also appears to produce action chains with seamless transitions between actions.

## Introduction

Many plan-based autonomous robot controllers generate chains of abstract actions in order to achieve complex, dynamically changing, and possibly interacting goals. To allow for plan-based control, the plan generation mechanisms are equipped with libraries of actions and causal models of these actions, specifying what it can achieve, and under which circumstances. By specifying these actions abstractly, they apply to a broad range of situations, reducing the search space for planning.

The advantages of this abstraction, however, come at a cost. Because planning systems consider actions as black boxes with performance independent of the prior and subsequent steps, the system cannot tailor the actions to the contexts of their execution. This often yields suboptimal behavior with abrupt transitions between actions, causing suboptimal performance. The resulting motion patterns are so characteristic for robots that people trying to imitate robotic behavior will do so by making abrupt movements between actions.

Let us illustrate these points using the autonomous robot soccer scenario depicted in Figure 1. To solve this task, the planner issues a three step plan, also shown in the figure. If the robot naively executes the first action (sub-figure 1b), it might arrive at the ball with the goal at its back, an unfortunate position from which to start dribbling towards the goal. The problem is that in the abstract view of the planner, being at the ball is considered sufficient for dribbling the ball and the dynamical state of the robot arriving at the ball is considered to be irrelevant for the dribbling action. What we would like the robot to do instead is to go to the ball *in order* to dribble it towards the goal afterwards. The robot should, as depicted in the sub-figure 1c, perform the first action sub-optimally in order to achieve a much better position for executing the second plan step.

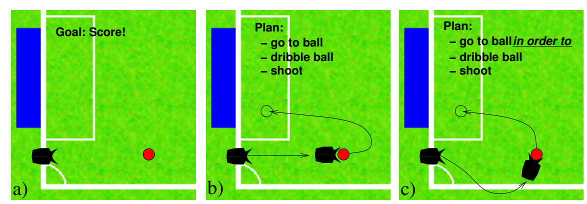


Figure 1: Alternative executions of the same plan

In this paper we propose a novel computational model for plan execution that enables the planner to keep its abstract action models and that optimizes action chains at execution time, shown in Figure 2. The basic idea of our approach is to learn performance models of abstract actions off-line from observed experience. Then at execution time, our system determines the set of parameters that are not set by the plan and therefore define the possible action executions. It then computes for each abstract action the parameterization such that the predicted performance of the action chain is optimal. This is done by refining the intermediate state between subsequent actions.

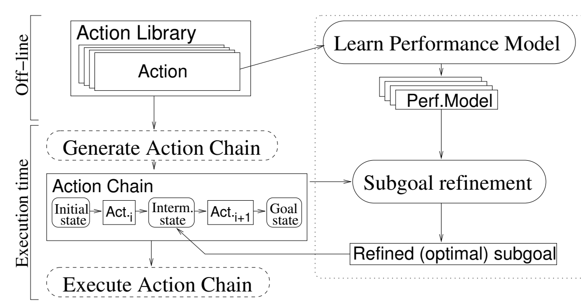


Figure 2: System Overview

## Learning performance models

To optimize action chains, we need performance models of each abstract action that predict the performance, e.g. time, given specific situations. The execution time of the `goToPose` action, which is based on computing a Bezier

*The work described in this paper was partially funded by the Deutsche Forschungsgemeinschaft in the SPP-1125.

curve and trying to follow it as closely as possible, depends on the distance (*dist*) and angle (*angle2dest*) to the destination, as well as the angle between the current orientation and the desired orientation at the destination (*angle@dest*).

The performance function for this action (`goToPose.perform`(*dist*,*angle2dest*,*angle@dest*)→*t*) is learned by model trees from observed experience acquired in a simulator, similar to (Belker *et al.* 2002). Model trees are a generalization of decision trees. They are functions that map continuous or nominal features to a continuous value. The function is learned from examples, by a piecewise partitioning of the feature space. A linear function is fitted to the data in each partition.

In Figure 3, we depict an example situation in which *dist* and *angle2dest* are 2.0m and 0°. The plots depict the predicted execution time for different angles of approach (*angle@dest*). The model tree's piecewise linear approximation is obvious in the Cartesian plot. The polar plot more clearly shows the dependency of predicted execution time on the angle of approach for the example situation. Note that the model has learned to predict performance for all situations the action can perform, and not just this specific situation.

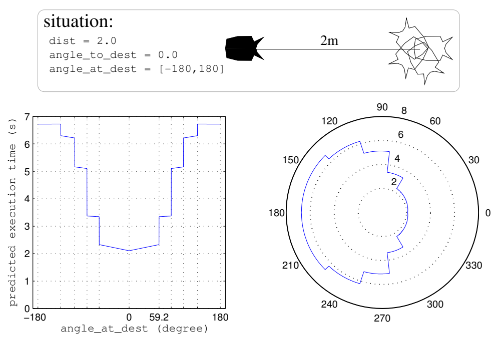


Figure 3: Temporal prediction with performance models

## Automatic subgoal refinement

The set of possible intermediate states between two actions is limited by the post-conditions of the first, and preconditions of the second action. The actual intermediate state simply arises after having executed the first action, as can be seen in Figure 1b. As it turns out, this state leads to suboptimal overall performance. From all possible intermediate states, our subgoal refinement system chooses the state that optimizes the predicted performance of the action chain.

In our example, the only variable free for optimizing is the angle of approach of the intermediate position. Our system automatically determines this by reasoning about the performance model (which variables influence performance), the pre- or post-conditions of the subsequent action (which variables are bound), and the current state of the world (which variables are fixed in the current state).

In Figure 4 the first two polar plots represent the performance of the two individual actions for different values of angle of approach. The overall performance is computed by adding those two, and is depicted in the third polar plot. The fastest time in the first polar plot is 2.1s, for angle of approach of 0.0°. However, the overall time is 7.5s. These values can be read directly from the polar plots. This value is not the optimum overall performance, which is actually 6.1s, as can be read from the third polar plot. Below the polar plots, the situation of Figure 1 is repeated, this time with the predicted performance for each action.

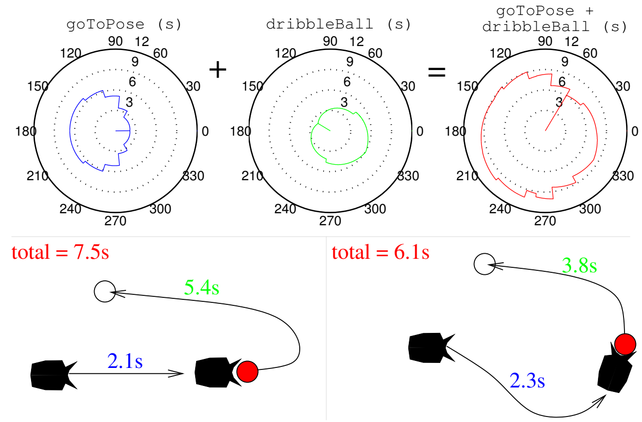


Figure 4: Computing the optimal intermediate goal.

## Results

To determine the influence of subgoal refinement on the overall performance of the action chain, we generated 1000 situations with random robot, ball and final goal positions. The robot executed each navigation task twice, once with subgoal refinement, and once without. The overall mean improvement is 12%. We have split these cases into those in which the subgoal refinement yielded a higher, equal or lower performance in comparison to not using refinement. This shows that the performance improved in 505 cases, and in these cases causes a 23% improvement. In 485 cases, there was no improvement. This is to be expected, as there are many situations in which the three positions are already optimally aligned (e.g. in a straight line), and subgoal refinement will have no effect. A small decrease in performance (6%) occurred in 10 cases.

## Conclusion and Future Work

On-line optimization of action chains allows the use of planning with abstract actions, without losing performance. Optimizing the action chain is done by refining under-specified intermediate goals, which requires no change in the planner or plan execution mechanisms. To predict the optimal overall performance, performance models of each individual abstract action are learned off-line and from experience, using model trees. It is interesting to see that requiring optimal performance can implicitly yield smooth transitions in robotic and natural domains, even though smoothness in itself is not an explicit goal. Applying subgoal refinement to the presented scenario yields good results. However, the computational models underlying the optimization are not specific to this scenario, or to robot navigation.

## References

T. Belker, M. Beetz, and A.B. Cremers. Learning action models for the improved execution of navigation plans. *Robotics and Autonomous Systems*, 38(3-4):137–148, 2002.

F. Stulp, M. Beetz. Optimized execution of action chains using learned performance models of abstract actions. *IJCAI*, 2005.